CLAIMS

What is claimed is:

1. A method of deactivating a process by a computer operating system, the method comprising:
initiating a process-wide deactivation operation;
determining whether threads of the process are currently suspendable; and
moving the threads of the process that are currently suspendable to a stopped state;
wherein the process-wide deactivation operation is called by outstanding threads of the process when the outstanding threads re-enter a kernel of the operating system.

2. The method of claim 1, wherein a thread is currently suspendable if the thread is stopped.

3. The method of claim 2, wherein a thread is also currently suspendable if the thread is sleeping interruptibly.

4. The method of claim 3, wherein a thread is also currently suspendable if the thread is interruptible and not currently running, and wherein the thread is removed from a run queue prior to moving the thread to the stopped state.

5. The method of claim 1, further comprising:
determining whether threads of the process are sleeping on memory; and
counting the threads of the process that are sleeping on memory to determine a number of memory sleepers.

6. The method of claim 5, further comprising:
calculating a sum of a number of threads in the stopped state and the number of memory sleepers.

7. The method of claim 6, further comprising:
determining if the sum is equal to a number of live threads in the process.

8. The method of claim 7, further comprising:
returning and indicating self-deactivation pending if the sum is unequal to the number of live threads.

9. The method of claim 7, further comprising:
moving the threads of the process that are sleeping on memory to the stopped state if the sum is equal to the number of live threads.

10. The method of claim 9, further comprising:
determining if the process is still deactivatable.

11. The method of claim 10, further comprising, if the process is still deactivatable, then finishing the process-wide deactivation operation and turning off a flag for the process.

12. The method of claim 10, if the process is no longer deactivatable, then un-suspending threads of the process and finishing the process-wide deactivation operation.

13. The method of claim 1, wherein zombie threads are not analyzed.

14. The method of claim 1, wherein the method is performed by code of a memory swapper of the operating system.

15. The method of claim 1, further comprising:
compelling a thread re-entering the kernel to enter the process deactivation operation.

16. The method of claim 15, wherein the process deactivation operation performs steps including:
moving the re-entering thread to the stopped state.

17. The method of claim 16, wherein the steps performed by the process deactivation operation further includes:
counting the threads of the process that are sleeping on memory to determine a number of memory sleepers;
calculating a sum of a number of threads in the stopped state and the number of memory sleepers; and
determining if the sum is equal to a number of live threads in the process.

18. The method of claim 17, wherein if the sum is equal, then the steps performed by the deactivation operation further includes:
un-suspending threads of the process and ending the process-wide deactivation operation if a kill command has been received.

19. The method of claim 17, wherein if the sum is equal, then the steps performed by the deactivation operation further includes:
un-suspending threads of the process and ending the process-wide deactivation operation if the process is no longer deactivatable.

20. The method of claim 17, wherein if the sum is equal, then the steps performed by the deactivation operation further includes, if no kill has been received and the process is still deactivatable, then completing deactivation of the process.

21. The method of claim 1, further comprising:
compelling a thread going to sleep for memory to check whether a deactivation is in progress for the thread's process.

22. The method of claim 1, wherein a deactivated process is reactivated by a procedure that includes bringing in associated memory regions, turning on a flag, and un-suspending threads of the process.

23. A multi-threaded operating system configured to deactivate a multi-threaded process by initiating a process-wide deactivation operation, determining whether threads of the process are currently suspendable, and moving the threads of the process that are currently suspendable to a stopped state, wherein the process-wide deactivation operation is called by outstanding threads of the process when the outstanding threads re-enter a kernel of the operating system's kernel.

24. A computer system having multi-threaded processes, the system comprising:

a process deactivation procedure which initiates a process-wide deactivation operation;

means for determining whether threads of the process are currently suspendable; and

means for moving the threads of the process that are currently suspendable to a stopped state;

wherein the process-wide deactivation operation is called by outstanding threads of the process when the outstanding threads re-enter a kernel of the operating system's kernel.